## 一，Tomcat相关配置

上面几乎没做什么特别的配置，大部分都使用了SpringBoot提供的默认的配置方式。有的时候我们可能需要有一些自定义的配置，比如Tomcat的配置，很简单，和[上上篇博客](http://blog.csdn.net/u012702547/article/details/53740047" \t "http://blog.csdn.net/u012702547/article/details/_blank)说的基本一致，有两种不同的配置方式：

### **在application.properties中配置**

直接在application.properties中进行配置即可，如下：

server.port=8081#配置服务器端口，默认为8080

server.session-timeout=1000000#用户回话session过期时间，以秒为单位

server.context-path=/index#配置访问路径，默认为/

server.tomcat.uri-encoding=UTF-8#配置Tomcat编码，默认为UTF-8

server.tomcat.compression=on#Tomcat是否开启压缩，默认为关闭

### **在代码中进行配置**

@Component

public class CustomServletContainer implements EmbeddedServletContainerCustomizer {

@Override

public void customize(ConfigurableEmbeddedServletContainer container) {

container.setPort(8080);

container.addErrorPages(new ErrorPage(HttpStatus.NOT\_FOUND,"/404.html"));

container.setSessionTimeout(10, TimeUnit.MINUTES);

}

}

自定义类实现 EmbeddedServletContainerCustomizer接口，然后设置端口、设置错误请求页面、设置会话超时时间等，大家注意这里的404页面放在src/main/resources/static文件夹下，有了这个之后，当我访问一个不存在的页面的时候就会跳转到404.html页面了。

### 二，SpringMVC相关配置

虽然Spring Boot默认的配置很多情况都可以满足我们的项目需求，可是有的时候我们可能还是会需要更加灵活的SpringMVC配置，这个时候我们只需要自定义类继承自WebMvcConfigurerAdapter，然后使用@Configuration和@EnableWebMvc注解，这样我们会完全屏蔽掉Spring Boot的默认配置，但是正常情况下我们可能只是希望在Spring Boot已有默认配置的基础上再添加一些配置即Spring Boot提供的默认配置和我自定义的配置并存的情况，这个也简单，只需要去掉@EnableWebMvc注解就行了。如下代码：

@Configuration

//@EnableWebMvc//无需使用该注解，否则会覆盖掉SpringBoot的默认配置值

public class WebMVCConfig extends WebMvcConfigurerAdapter {

@Override

public void addViewControllers(ViewControllerRegistry registry) {

registry.addViewController("/hello").setViewName("/hello");

}

}

### **三，actuator监控**

导包spring-boot-starter-actuator；无法启动actuator时，重新导入，maven--update。

1. 可以加入依赖spring-boot-starter-security来保证actuator暴露接口的安全性，可以通过 -u 'user:password' 方式来访问basic auth。
2. 如果项目依赖的是springmvc框架，并且基础的配置文件是 application.yaml的话，可以增加 application.properties 文件来配置安全性的配置.
3. 如果加入了security依赖，则所有的接口默认都需要被验证，默认写在controller里通过@RestMapping对外暴露的接口也会被权限保护，访问时会要你输入账号密码。如果只想 /admin路径下的请求进行验证，则需要加入配置

security.basic.enabled=true

security.basic.path=/admin

security.user.name=admin

security.user.password=password

1. 如果项目依赖的是非springmvc框架的话， 需要在依赖中加入mvc的依赖

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

</dependency>

1. 如果management.security.enabled的值是false的话，除开health接口还依赖endpoints.health.sensitive的配置外，其他接口都不需要输入用户名和密码了。
2. actuator暴露的health接口权限是由两个配置： management.security.enabled  和 endpoints.health.sensitive组合的结果进行返回的。

| management.security.enabled | endpoints.health.sensitive | Unauthenticated | Authenticated |
| --- | --- | --- | --- |
| false | false | Full content | Full content |
| false | true | Status only | Full content |
| true | false | Status only | Full content |
| true | true | No content | Full content |

1. actuator组件里面除开上面提到的metrics和health接口以外，还有很多默认的其他接口，如果它默认的接口不能满足你的需求的话，还可以通过继承它的 AbstractEndpoint类来实现自己的Endpoint
2. 配置文件例子

security.basic.enabled=true

security.basic.path=/admin #针对/admin路径进行认证

security.user.name=admin #认证使用的用户名

security.user.password=password #认证使用的密码

management.security.roles=SUPERUSER

management.port=11111 #actuator暴露接口使用的端口，和api接口使用的端口进行分离

management.context-path=/admin #actuator暴露接口的前缀 management.security.enabled=true #actuator是否需要安全保证 endpoints.metrics.sensitive=false #actuator的metrics接口是否需要安全保证 endpoints.metrics.enabled=true

endpoints.health.sensitive=false #actuator的health接口是否需要安全保证

endpoints.health.enabled=true

1. 映射路径，默认情况下：/beans---> /actuator/beans

修改：

management.endpoints.web.base-path=/

management.endpoints.web.path-mapping.health=healthcheck